UNIVERSIDAD SIGLO 21

Universidad siglo 21 Licenciatura en Informática
Seminario de Práctica.

ARANDA, Cristian – AP4.PDF

Sistema de Gestión de Ventas y Stock para Librería de Artículos Escolares

Índice

2. Introducción

En el contexto actual de transformación digital, los pequeños y medianos comercios enfrentan el desafío de modernizar sus procesos para mejorar la eficiencia operativa y la atención al cliente. Las librerías y papelerías, especialmente aquellas dedicadas a la venta de artículos escolares, requieren herramientas tecnológicas que les permitan gestionar sus ventas de manera ágil, precisa y organizada.

Este proyecto propone el desarrollo de un sistema informático destinado a la gestión de ventas de artículos escolares, orientado a optimizar el control de productos, automatizar el registro de transacciones y facilitar la generación de reportes comerciales. La implementación de este sistema busca reducir errores humanos, mejorar la administración del stock y brindar una experiencia más eficiente tanto para el vendedor como para el cliente.

El sistema estará diseñado con una interfaz intuitiva, adaptable a distintos entornos comerciales, y permitirá el acceso a funcionalidades clave como la carga de productos, la realización de ventas, el seguimiento del inventario y la obtención de estadísticas de rendimiento. De esta manera, se pretende contribuir al fortalecimiento de la gestión comercial en el rubro escolar, promoviendo el uso de soluciones tecnológicas accesibles y efectivas.

3. Justificación

La gestión eficiente de ventas en comercios dedicados a la distribución de artículos escolares representa un factor clave para su sostenibilidad y crecimiento. En muchos casos, estos negocios operan con métodos manuales o sistemas rudimentarios que dificultan el control del inventario, la trazabilidad de las transacciones y la generación de reportes útiles para la toma de decisiones.

La falta de automatización en estos procesos puede derivar en errores frecuentes, pérdida de información, demoras en la atención al cliente y una baja capacidad de respuesta ante períodos de alta demanda, como el inicio del ciclo lectivo. Además, la ausencia de herramientas tecnológicas limita la posibilidad de analizar el comportamiento de ventas, identificar productos más solicitados y planificar estrategias comerciales efectivas.

Por estas razones, se justifica el desarrollo de un sistema informático que permita gestionar de manera integral las ventas de artículos escolares. Este sistema no solo contribuirá a mejorar la organización interna del negocio, sino que también facilitará la atención al cliente, optimizará el control del stock y proporcionará información valiosa para la administración comercial. La implementación de esta solución tecnológica representa una oportunidad concreta para modernizar el sector y fortalecer la competitividad de los pequeños comerciantes.

4. Definiciones del proyecto y del sistema

Definición del proyecto: El proyecto consiste en el análisis, diseño y desarrollo de un sistema de información que permita la gestión integral de ventas y stock en una librería de artículos escolares.

Objetivo general:

- Desarrollar un sistema de gestión de ventas y control de stock para una librería de artículos escolares que permita centralizar la información, mejorar la eficiencia operativa y brindar un mejor servicio al cliente.

Objetivos específicos:

1. Automatizar el registro de productos escolares con sus características (nombre, marca, precio, cantidad disponible).
2. Controlar de manera precisa el inventario y las actualizaciones de stock en tiempo real.
3. Facilitar el registro de ventas y la emisión de comprobantes.
4. Generar reportes de ventas y stock para apoyar la toma de decisiones del administrador.
5. Implementar un sistema de roles que distinga entre administrador y vendedor.
6. Reducir errores y demoras ocasionados por la gestión manual actual.

Definición del sistema:
El sistema será una aplicación informática que permitirá registrar productos escolares, gestionar inventario, realizar ventas, emitir comprobantes y generar reportes de control, accesible tanto para administradores como para vendedores.

Alcance del sistema:

- Gestión de productos (altas, bajas, modificaciones, consulta).
- Control de stock en tiempo real.
- Registro de ventas y emisión de comprobantes.
- Generación de reportes de ventas y stock.
- Administración de usuarios con diferentes roles (administrador y vendedor).

Límites Del Sistema:

- El sistema se limitará a la gestión de productos y ventas dentro de la librería.
- No contemplará integración con sistemas contables externos.
- No incluirá gestión de compras a proveedores.
- No tendrá funcionalidades de comercio electrónico (ventas online).

Restricciones:

- El sistema debe operar dentro del horario y red local de la librería.
- El acceso estará limitado únicamente a usuarios registrados.
- Se utilizarán dispositivos estándar de la librería (PCs de escritorio).
- Se priorizará la facilidad de uso para vendedores sin formación técnica avanzada.

5. Elicitación

Para definir los requerimientos funcionales y no funcionales del sistema, se llevaron a cabo entrevistas semiestructuradas con propietarios y empleados de librerías locales, así como observaciones directas del proceso de venta en comercios del rubro. También se aplicaron encuestas breves a clientes habituales para conocer sus expectativas respecto a la atención y disponibilidad de productos.

A partir de estas actividades de recolección de información, se identificaron las siguientes necesidades clave:

- Registro rápido y sencillo de productos escolares, con posibilidad de clasificar por categoría (útiles, mochilas, cuadernos, etc.).
- Control automático del stock, con alertas ante niveles bajos de inventario.
- Registro de ventas con cálculo automático de totales y aplicación de descuentos.
- Generación de reportes diarios, semanales y mensuales para el análisis comercial.
- Interfaz amigable que permita operar el sistema sin conocimientos técnicos avanzados.
- Seguridad en el manejo de datos y respaldo periódico de la información.

5.1 **Técnicas utilizadas para la elicitación**

Estos requerimientos fueron priorizados en función de su impacto en la operación del negocio y su viabilidad técnica, y constituyen la base para el diseño del sistema propuesto.

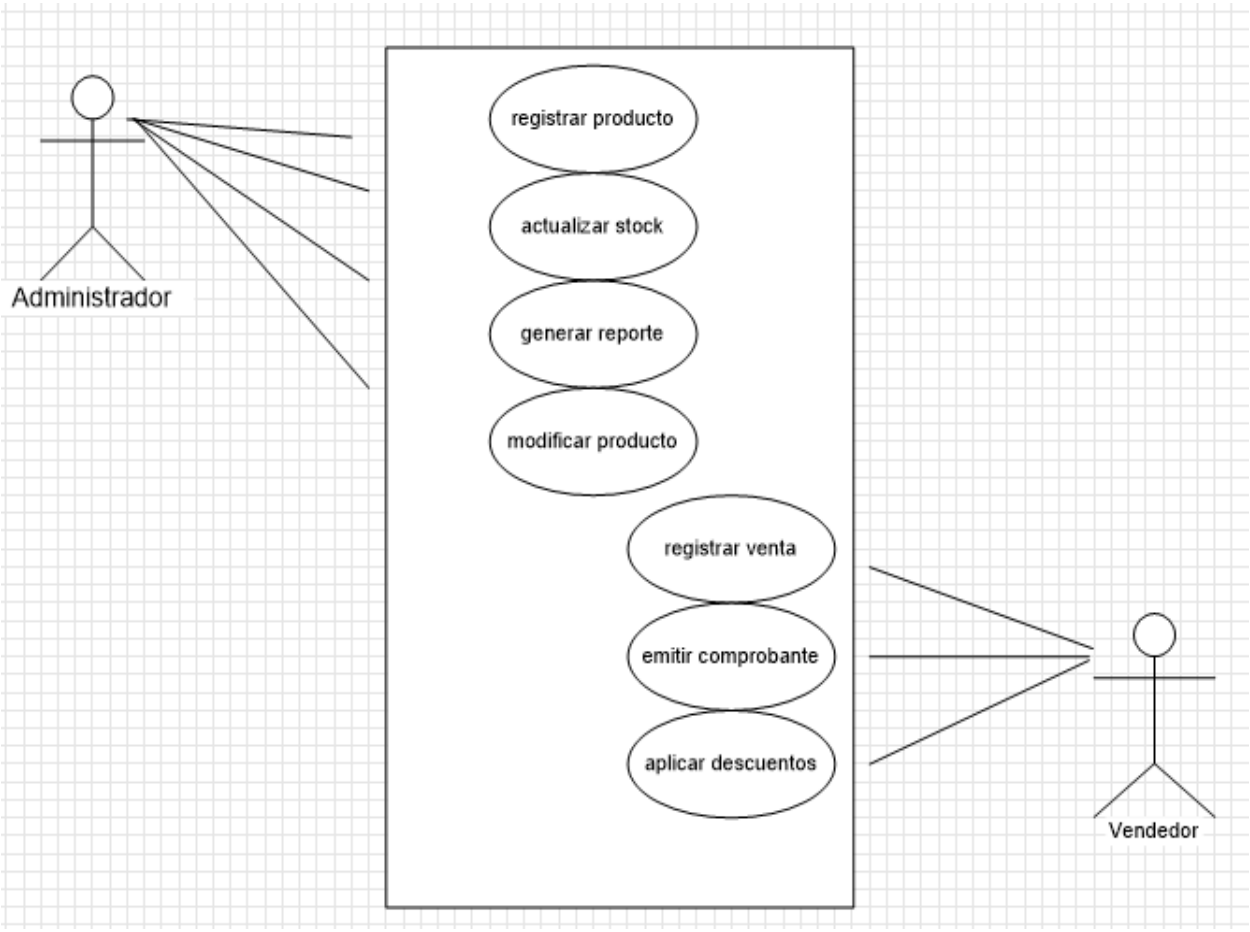
- **Entrevistas:** realizadas con el dueño de la librería y con dos vendedores.
- **Observación directa:** se observaron las tareas cotidianas durante la atención al cliente y el manejo de stock.
- **Análisis documental:** revisión de planillas de control de stock y comprobantes de ventas manuales.

5.2 Actores identificados

- **Administrador:** Responsable de la gestión de stock, consulta de reportes y administración de usuarios.
- **Vendedor:** Encargado de registrar ventas y emitir comprobantes.
- **Cliente:** Usuario final que adquiere artículos escolares (interacción indirecta).

Caso de uso	Actor	Descripción breve
Registrar producto	Administrador	Carga nuevos artículos escolares al sistema
Modificar producto	Administrador	Actualiza información de productos existentes
Consultar stock	Administrador	Verifica cantidad disponible de productos
Realizar venta	Vendedor	Selecciona productos y finaliza la transacción
Aplicar descuento	Vendedor	Aplica descuentos sobre el total de la venta
Emitir comprobante	Vendedor	Genera comprobante de la venta realizada
Generar reporte de ventas	Administrador	Obtiene estadísticas por día, semana o mes

Diagrama de caso de uso



6. Conocimiento del negocio

El presente proyecto se enmarca en el contexto de una librería o papelería dedicada a la comercialización de artículos escolares, tales como útiles, mochilas, cuadernos, carpetas, lápices, entre otros. Este tipo de comercio presenta una dinámica estacional marcada, con picos de demanda durante el inicio del ciclo lectivo y fechas especiales como el Día del Estudiante o promociones escolares.

El negocio opera bajo un esquema de atención presencial, donde el vendedor registra las ventas en tiempo real y gestiona el inventario de forma manual o semi automatizada. El volumen de productos manejados puede superar los 500 ítems, organizados por categorías y marcas, lo que exige un control preciso del stock y una rápida capacidad de respuesta ante consultas de los clientes.

Actualmente, la gestión de ventas se realiza mediante planillas físicas o herramientas básicas como hojas de cálculo, lo que limita la trazabilidad de las transacciones, dificulta la generación de reportes y aumenta el riesgo de errores humanos. Además, la falta de integración entre el registro de ventas y el control de inventario impide una visión clara del rendimiento comercial.

El sistema propuesto busca responder a estas necesidades mediante la automatización de los procesos clave del negocio: registro de productos, ejecución de ventas, actualización del stock y generación de reportes. De esta manera, se pretende mejorar la eficiencia operativa, reducir los errores administrativos y brindar al comerciante una herramienta tecnológica accesible para la toma de decisiones informadas.

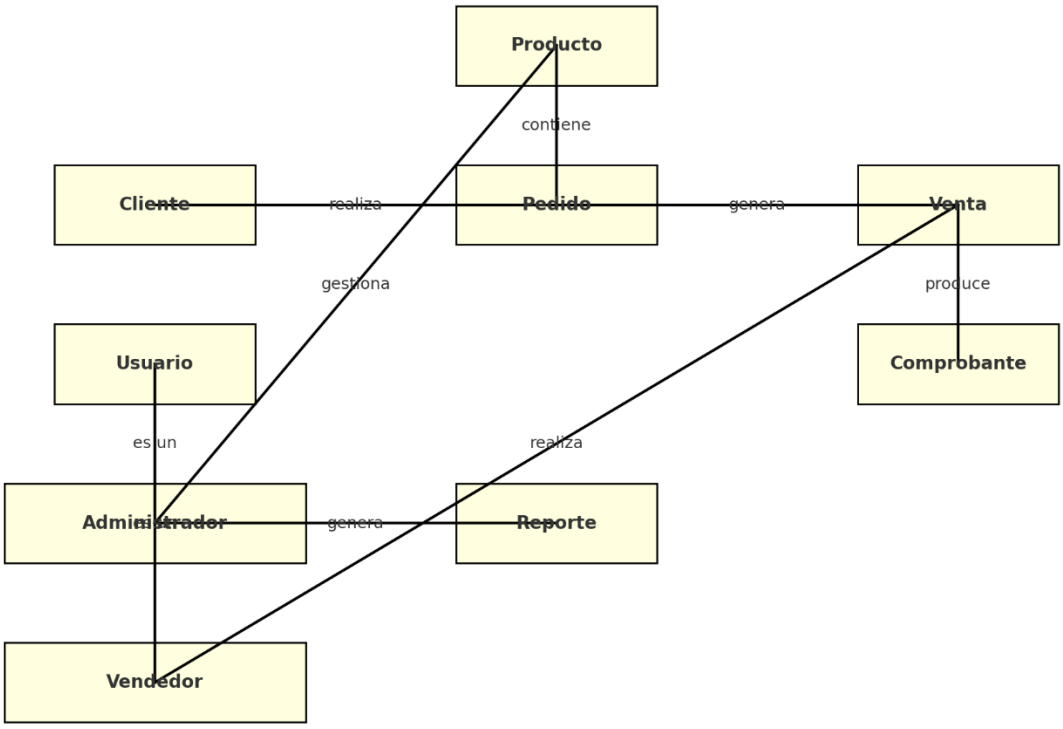
Conceptos principales

- **Producto**: representa cada artículo escolar (nombre, precio, stock).
- **Cliente**: quien realiza pedidos y compras.
- **Pedido**: solicitud de uno o varios productos.
- **Venta**: operación registrada cuando el pedido es confirmado.
- **Comprobante**: factura o ticket emitido al cliente.
- **Usuario**: representa a los actores que interactúan con el sistema (administrador, vendedor).
- **Reporte**: documento generado por el administrador.

Relaciones

- Un **Cliente** puede realizar muchos **Pedidos**.
- Un **Pedido** contiene uno o varios **Productos**.
- Una **Venta** se genera a partir de un **Pedido** confirmado.
- Una **Venta** produce un **Comprobante**.
- Un **Administrador** (tipo de Usuario) gestiona productos y genera reportes.
- Un **Vendedor** (tipo de Usuario) realiza ventas y emite comprobantes.

Diagrama de Dominio - Sistema de Gestión de Ventas de Artículos Escolares



Descripción de Entidades del Dominio

1. **Producto**
   - Representa los artículos escolares disponibles en la librería.
   - Atributos principales: nombre, categoría, precio, cantidad en stock.
   - Relación: puede estar asociado a múltiples pedidos.
2. **Cliente**
   - Persona que consulta, solicita y compra productos en la librería.
   - Atributos principales: nombre, apellido, DNI, contacto.
   - Relación: puede realizar varios pedidos.
3. **Pedido**
   - Conjunto de productos solicitados por un cliente.
   - Atributos principales: número de pedido, fecha, estado.
   - Relación: un pedido contiene uno o más productos y genera una venta.
4. **Venta**
   - Operación registrada al concretarse un pedido.
   - Atributos principales: número de venta, fecha, monto total.
   - Relación: cada venta produce un comprobante.
5. **Comprobante**
   - Documento emitido como constancia de la venta (factura o ticket).
   - Atributos principales: número de comprobante, fecha, tipo, monto.
   - Relación: está asociado a una venta.
6. **Usuario**

- Representa a las personas que interactúan con el sistema interno.
- Atributos principales: usuario, contraseña, rol.
- Relación: puede ser **Administrador** o **Vendedor**.

7. **Administrador (especialización de Usuario)**
   - Encargado de la gestión interna del negocio.
   - Responsabilidades: alta, baja y modificación de productos, control de stock, generación de reportes.
8. **Vendedor (especialización de Usuario)**
   - Encargado de la atención al cliente en el punto de venta.
   - Responsabilidades: registrar ventas, aplicar descuentos, emitir comprobantes.
9. **Reporte**
   - Documento generado por el administrador para el control de gestión.
   - Atributos principales: tipo de reporte (ventas, stock), período de tiempo, fecha de generación.
   - Relación: generado por el administrador.

7. Propuesta de solución

A partir del relevamiento de procesos y necesidades del negocio, se propone el desarrollo de un sistema informático que automatice y optimice la gestión de ventas en comercios dedicados a la comercialización de artículos escolares. El sistema estará orientado a mejorar la eficiencia operativa, reducir errores administrativos y facilitar la toma de decisiones mediante el acceso a información confiable y actualizada.

La solución contempla el diseño e implementación de una aplicación con arquitectura cliente-servidor, desarrollada con tecnologías web y una base de datos relacional. El sistema incluirá los siguientes módulos funcionales:

Módulos del sistema

- **Gestión de productos** Permite registrar, modificar y eliminar artículos escolares, clasificarlos por categoría, asignar precios y controlar el stock disponible.
- **Gestión de ventas** Facilita la realización de transacciones, con selección de productos, cálculo automático de totales, aplicación de descuentos y emisión de comprobantes.
- **Control de stock** Actualiza automáticamente el inventario tras cada venta, con alertas ante niveles bajos y posibilidad de consultar el estado de productos en tiempo real.
- **Reportes comerciales** Genera informes diarios, semanales y mensuales sobre ventas, productos más vendidos, ingresos totales y evolución del negocio.
- **Seguridad y respaldo** Implementa mecanismos de autenticación, control de acceso por roles y respaldo periódico de la base de datos para garantizar la integridad de la información.

Tecnologías sugeridas

- **Frontend**: HTML, CSS, JavaScript
- **Backend**: PHP, Python o Node.js
- **Base de datos**: MySQL
- **Herramientas de modelado**: UML para casos de uso y clases, BPMN para procesos

Esta solución busca ser escalable, adaptable a distintos tamaños de negocio, y fácil de usar para usuarios sin conocimientos técnicos avanzados. Su implementación permitirá al comerciante mejorar la organización interna, agilizar la atención al cliente y disponer de información estratégica para la gestión comercial.

8- Inicio del análisis

Requerimientos

El presente apartado tiene como objetivo describir de manera estructurada los requerimientos del sistema informático propuesto, orientado a la gestión de ventas de artículos escolares en comercios minoristas. Los requerimientos representan las funcionalidades y características que el sistema debe cumplir para responder adecuadamente a las necesidades del negocio, garantizando su utilidad, eficiencia y calidad.

La identificación de estos requerimientos se realizó mediante un proceso de elicitación que incluyó entrevistas con los propietarios y empleados del comercio, observación directa de los procesos operativos, y análisis de documentación existente. A partir de esta información, se definieron los requerimientos funcionales, que describen las acciones específicas que el sistema debe realizar, y los requerimientos no funcionales, que establecen criterios de calidad, rendimiento, seguridad y usabilidad.

La correcta definición de los requerimientos constituye una base fundamental para el diseño, desarrollo y validación del sistema, asegurando que las funcionalidades implementadas respondan a los objetivos del proyecto y a las expectativas de los usuarios finales.

Requerimientos Funcionales (RF)

ID Requerimiento	Descripción
RF1	Registrar nuevos productos escolares
RF2	Modificar datos de productos existentes
RF3	Eliminar productos del catálogo
RF4	Mostrar el stock disponible en tiempo real
RF5	Registrar una venta

ID Requerimiento	Descripción
RF6	Aplicar descuentos en las ventas
RF7	Calcular el monto total automáticamente
RF8	Emitir comprobantes de venta
RF9	Consultar el catálogo de productos
RF10	Realizar pedidos
RF11	Enviar comprobante al cliente luego de la compra
RF12	Generar reportes de ventas
RF13	Generar reportes de stock bajo
RF14	Autenticación de administradores y vendedores
RF15	Restringir acceso a funcionalidades según el rol

Requerimientos no Funcionales

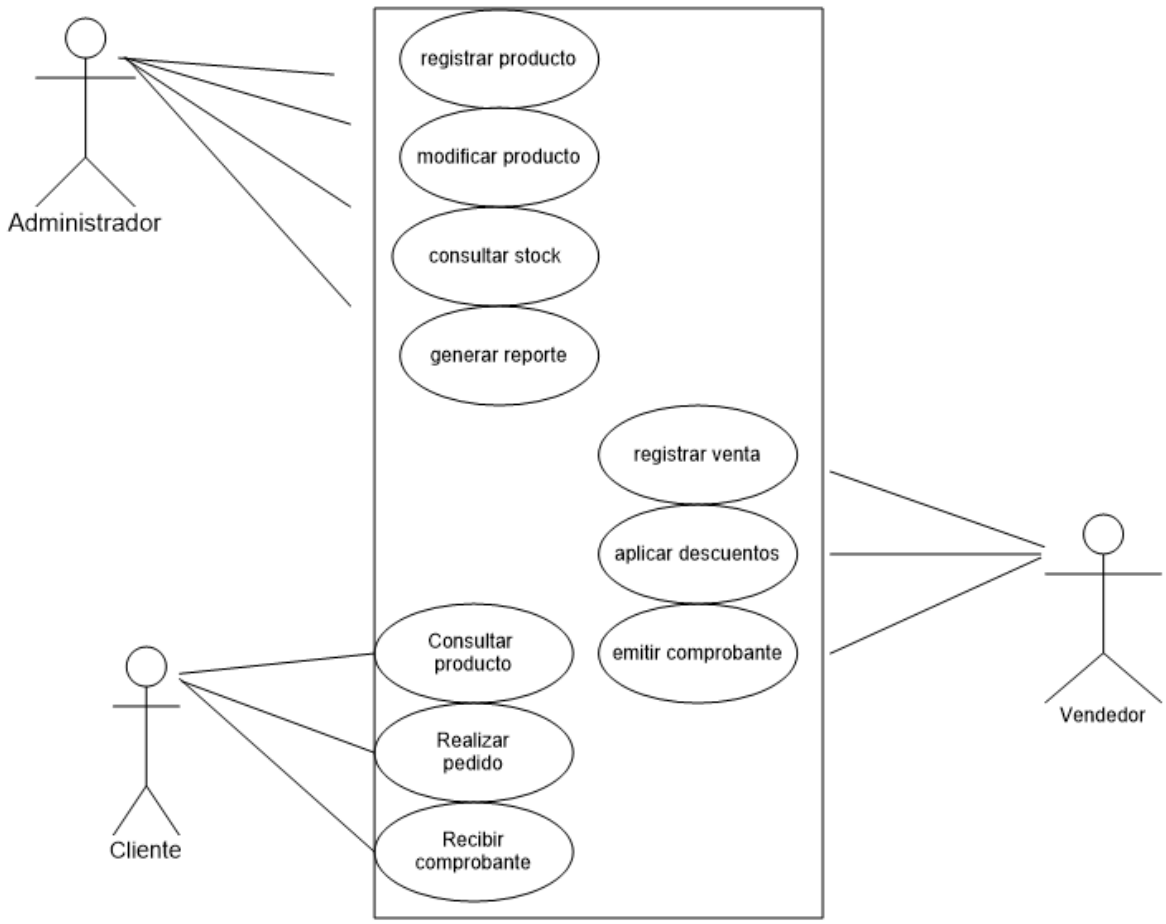
RNF1	Procesar una venta en menos de 5 segundos
RNF2	Soportar hasta 20 usuarios concurrentes
RNF3	Interfaz intuitiva y fácil de usar
RNF4	Disponibilidad en idioma español
RNF5	Autenticación obligatoria con usuario y contraseña
RNF6	Encriptación de contraseñas en la base de datos
RNF7	Disponibilidad del 95% del tiempo laboral
RNF8	Ejecución en Windows y Linux
RNF9	Posibilidad de agregar nuevos reportes sin afectar la arquitectura
RNF10	Código documentado y estructurado

Diagrama de caso de uso

Como parte del análisis funcional del sistema de gestión de ventas de artículos escolares, se identificaron los principales casos de uso que representan las interacciones entre los actores del sistema y las funcionalidades que este debe ofrecer. Los casos de uso permiten modelar el comportamiento esperado del sistema desde la perspectiva del usuario, facilitando la comprensión de los requerimientos y guiando el diseño posterior.

Actores involucrados

- **Administrador**: Encargado de gestionar productos, consultar stock y generar reportes.
- **Vendedor**: Responsable de realizar ventas, aplicar descuentos y emitir comprobantes.
- **Cliente** (opcional): Puede consultar productos o realizar pedidos si el sistema lo permite.



Descripción de casos de uso

Caso de Uso 1: Registrar producto

- **Actor:** Administrador
- **Objetivo:** Incorporar un nuevo artículo escolar al inventario.
- **Flujo básico:**

1. El administrador selecciona la opción “Registrar producto”.
2. Ingresa datos del producto (nombre, categoría, precio, cantidad inicial).
3. El sistema valida la información.
4. El producto queda registrado en la base de datos.

- **Resultado esperado:** El nuevo producto queda disponible para consultas y ventas.

Caso de Uso 2: Modificar producto

- **Actor:** Administrador
- **Objetivo:** Actualizar la información de un producto existente.
- **Flujo básico:**
  1. El administrador selecciona “Modificar producto”.
  2. Busca el producto en la lista.
  3. Actualiza datos (precio, stock, descripción).
  4. El sistema guarda los cambios.
- **Resultado esperado:** La información del producto queda actualizada en el sistema.

Caso de Uso 3: Consultar stock

- **Actor:** Administrador
- **Objetivo:** Verificar el inventario actual.
- **Flujo básico:**
  1. El administrador solicita un reporte de stock.
  2. El sistema muestra los productos con cantidades disponibles.
- **Resultado esperado:** El administrador tiene una visión clara de las existencias.

Caso de Uso 4: Generar reportes

- **Actor:** Administrador
- **Objetivo:** Obtener información consolidada de ventas o stock.
- **Flujo básico:**
  1. El administrador selecciona el tipo de reporte (ventas, stock).
  2. Define período de tiempo.
  3. El sistema genera el reporte en pantalla o exportable.
- **Resultado esperado:** El administrador obtiene un documento de gestión confiable.

Caso de Uso 5: Realizar venta

- **Actor:** Vendedor
- **Objetivo:** Registrar la venta de productos a un cliente.
- **Flujo básico:**
  1. El vendedor selecciona los productos que el cliente desea comprar.
  2. El sistema descuenta las cantidades del stock.
  3. El sistema calcula el total.

- **Resultado esperado:** La venta queda registrada en el sistema.

Caso de Uso 6: Aplicar descuento

- **Actor:** Vendedor
- **Objetivo:** Aplicar un descuento a la venta.
- **Flujo básico:**
    1. El vendedor selecciona el pedido.
    2. Ingresa el porcentaje de descuento autorizado.
    3. El sistema recalcula el total.
- **Resultado esperado:** El precio final se ajusta correctamente.

Caso de Uso 7: Emitir comprobante

- **Actor:** Vendedor
- **Objetivo:** Generar el documento de venta.
- **Flujo básico:**
    1. El vendedor confirma la venta.
    2. El sistema emite el comprobante (ticket o factura).
    3. Se entrega al cliente.
- **Resultado esperado:** El cliente recibe su comprobante de compra.

Caso de Uso 8: Consultar productos

- **Actor:** Cliente
- **Objetivo:** Revisar los artículos disponibles.
- **Flujo básico:**
    1. El cliente solicita la lista de productos.
    2. El sistema muestra los artículos con precios y disponibilidad.
- **Resultado esperado:** El cliente puede conocer la oferta de productos.

Caso de Uso 9: Realizar pedido

- **Actor:** Cliente
- **Objetivo:** Solicitar productos a la librería.
- **Flujo básico:**
    1. El cliente selecciona los productos deseados.
    2. Define cantidades.
    3. El pedido queda registrado en el sistema.
- **Resultado esperado:** El cliente genera un pedido válido.

Caso de Uso 10: Recibir comprobante

- **Actor:** Cliente
- **Objetivo:** Obtener un comprobante de la compra.
- **Flujo básico:**
  1. El vendedor confirma la venta.
  2. El sistema emite el comprobante.
  3. El cliente lo recibe.
- **Resultado esperado:** El cliente dispone de un documento que respalda la transacción.

Tabla de Casos de Uso – Sistema de Gestión de Ventas de Artículos Escolares

ID	Caso de Uso	Actor	Descripción	Precondiciones	Flujo Principal	Flujos Alternativos
CU1	Registrar producto	Administrador	Permite registrar un nuevo producto en el sistema.	Administrador autenticado.	1. Selecciona *Registrar producto*. 2. Ingresa datos (nombre, precio, stock). 3. El sistema valida y guarda. 4. Muestra confirmación.	3a. Datos incompletos → mensaje de error.
CU2	Modificar producto	Administrador	Permite modificar datos de productos ya registrados.	Administrador autenticado.	1. Selecciona producto. 2. Ingresa cambios. 3. El sistema actualiza los datos. 4. Confirma modificación.	2a. Producto inexistente → error.
CU3	Consultar stock	Administrador	Muestra el stock disponible.	Administrador autenticado.	1. Selecciona *Consultar stock*. 2. El sistema muestra inventario actualizado.	—

ID	Caso de Uso	Actor	Descripción	Precondiciones	Flujo Principal	Flujos Alternativos
CU4	Generar reportes	Administrador	Obtiene reportes de ventas y stock.	Administrador autenticado.	1. Selecciona *Generar reporte*. 2. Ingresa criterios. 3. El sistema procesa y genera reporte. 4. Se presenta en pantalla/exporta.	3a. Sin datos en rango → aviso.
CU5	Realizar venta	Vendedor	Permite registrar una venta de productos.	Vendedor autenticado.	1. Selecciona *Nueva venta*. 2. Escoge productos y cantidades. 3. El sistema calcula total. 4. Confirma la venta. 5. El sistema actualiza stock y registra.	2a. Stock insuficiente → alerta, no permite continuar.
CU6	Aplicar descuento	Vendedor	Permite aplicar descuentos a una venta.	Venta en curso.	1. Vendedor selecciona opción *Aplicar descuento*. 2. Ingresa porcentaje/monto. 3. El sistema recalcula el total.	2a. Descuento no válido → error.
CU7	Emitir comprobante	Vendedor	Genera y entrega comprobante al cliente.	Venta registrada.	1. El sistema genera comprobante. 2. Se imprime o envía digitalmente. 3. El vendedor entrega al cliente.	2a. Falla impresora → comprobante disponible en PDF.
CU8	Consultar productos	Cliente	Permite visualizar catálogo disponible.	Cliente registrado.	1. Accede a *Consultar productos*. 2. El sistema muestra lista con precios y disponibilidad.	2a. Producto sin stock → “No disponible”.

ID	Caso de Uso	Actor	Descripción	Precondiciones	Flujo Principal	Flujos Alternativos
CU9	Realizar pedido	Cliente	Permite al cliente realizar un pedido.	Cliente registrado.	1. Cliente selecciona productos. 2. Confirma pedido. 3. El sistema registra el pedido y actualiza stock.	2a. Stock insuficiente → mensaje de error.
CU10	Recibir comprobante	Cliente	Permite recibir comprobante de su compra.	Venta registrada.	1. El sistema envía/entrega comprobante al cliente. 2. Cliente recibe documento.	1a. Fallo de envío digital → comprobante disponible en local.

Tabla de trazabilidad de requerimiento de casos de uso

ID Requerimiento	Descripción	Caso de Uso Asociado
RF1	Registrar nuevos productos escolares	Registrar producto
RF2	Modificar datos de productos existentes	Modificar producto
RF3	Eliminar productos del catálogo	Modificar producto
RF4	Mostrar el stock disponible en tiempo real	Consultar stock
RF5	Registrar una venta	Realizar venta
RF6	Aplicar descuentos en las ventas	Aplicar descuento
RF7	Calcular el monto total automáticamente	Realizar venta
RF8	Emitir comprobantes de venta	Emitir comprobante
RF9	Consultar el catálogo de productos	Consultar productos
RF10	Realizar pedidos	Realizar pedido
RF11	Enviar comprobante al cliente luego de la compra	Recibir comprobante
RF12	Generar reportes de ventas	Generar reportes
RF13	Generar reportes de stock bajo	Generar reportes
RF14	Autenticación de administradores y vendedores	Inicio de sesión (implícito, no dibujado en el diagrama)

ID Requerimiento	Descripción	Caso de Uso Asociado
RF15	Restringir acceso a funcionalidades según el rol	Todos los casos de uso (control de roles)
RNF1	Procesar una venta en menos de 5 segundos	Realizar venta
RNF2	Soportar hasta 20 usuarios concurrentes	Todos los casos de uso
RNF3	Interfaz intuitiva y fácil de usar	Todos los casos de uso
RNF4	Disponibilidad en idioma español	Todos los casos de uso
RNF5	Autenticación obligatoria con usuario y contraseña	Inicio de sesión
RNF6	Encriptación de contraseñas en la base de datos	Inicio de sesión
RNF7	Disponibilidad del 95% del tiempo laboral	Todos los casos de uso
RNF8	Ejecución en Windows y Linux	Todos los casos de uso
RNF9	Posibilidad de agregar nuevos reportes sin afectar la arquitectura	Generar reportes
RNF10	Código documentado y estructurado	Todos los casos de uso

9. Etapa de Análisis

El análisis permitió comprender en profundidad las necesidades de la librería de artículos escolares y transformar dichas necesidades en requerimientos técnicos.
Durante esta fase se identificaron los actores principales del sistema (Administrador, Vendedor y Cliente), los casos de uso más relevantes (gestión de productos, ventas, reportes y consultas de clientes) y se documentaron los requerimientos funcionales y no funcionales.

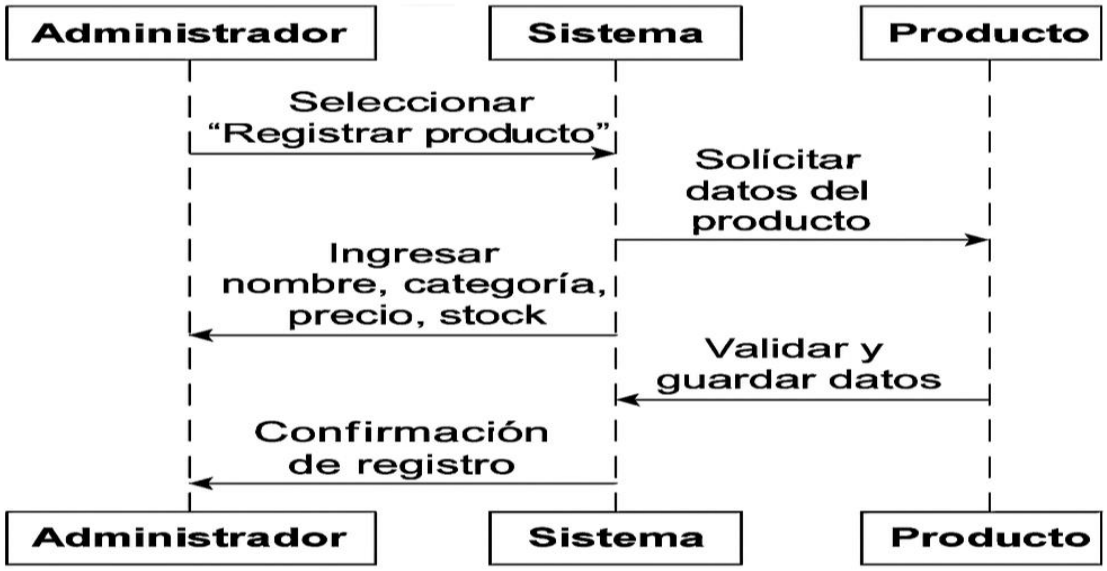
Asimismo, se delimitaron el alcance y los límites del sistema, estableciendo que la solución se enfocará en la gestión de ventas, control de stock y generación de reportes, dejando fuera aspectos como la logística de proveedores o la contabilidad avanzada.

El resultado de esta etapa fue un modelo conceptual del sistema representado en diagramas de secuencia y de casos de uso, que sirvió como base para el diseño posterior.

Diagramas de Secuencia – Sistema de Gestión de Ventas de Artículos Escolares

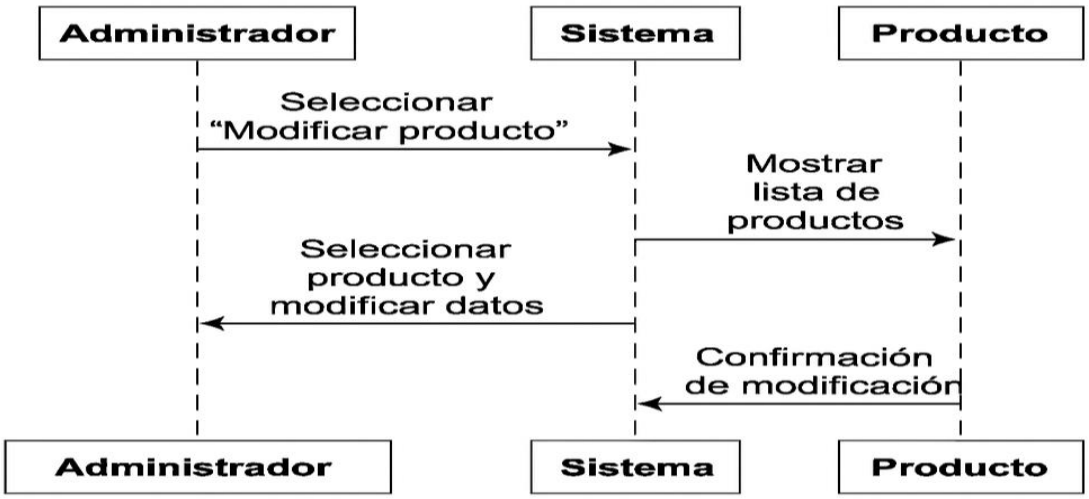
CU01- Registrar Producto

Diagrama de secuencia correspondiente al caso de uso: Registrar Producto.



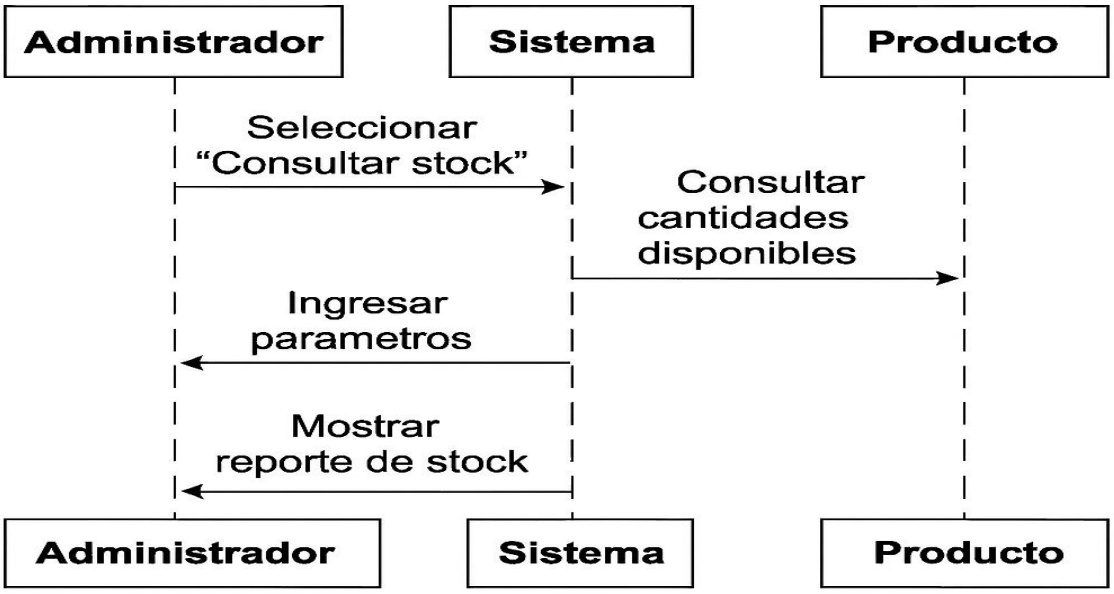
CU02- Modificar Producto

Diagrama de secuencia correspondiente al caso de uso: Modificar Producto.



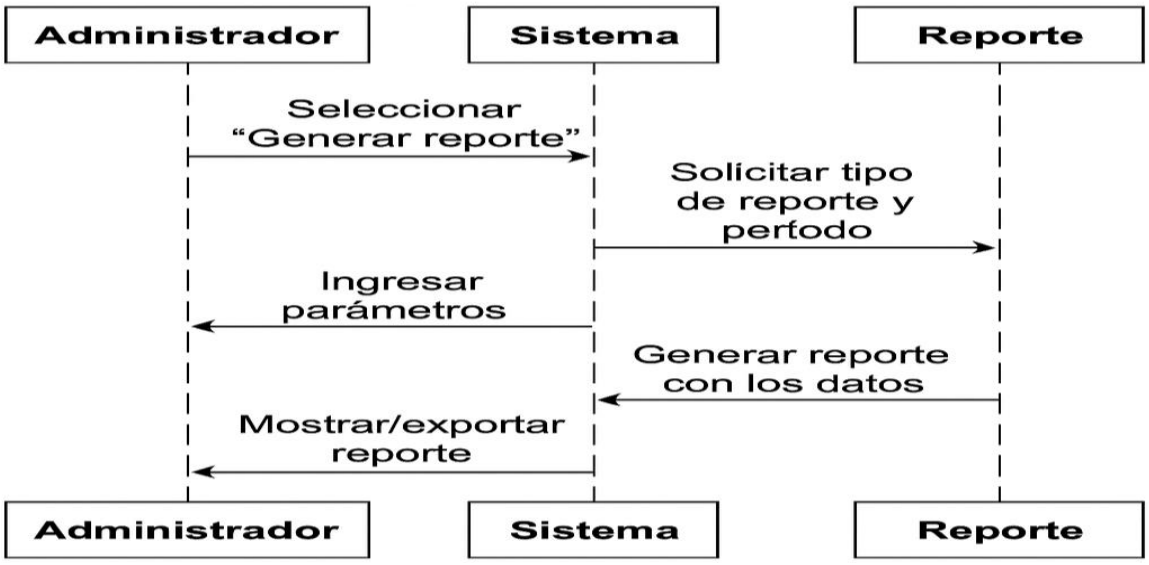
CU03- Consultar Stock

Diagrama de secuencia correspondiente al caso de uso: Consultar Stock.



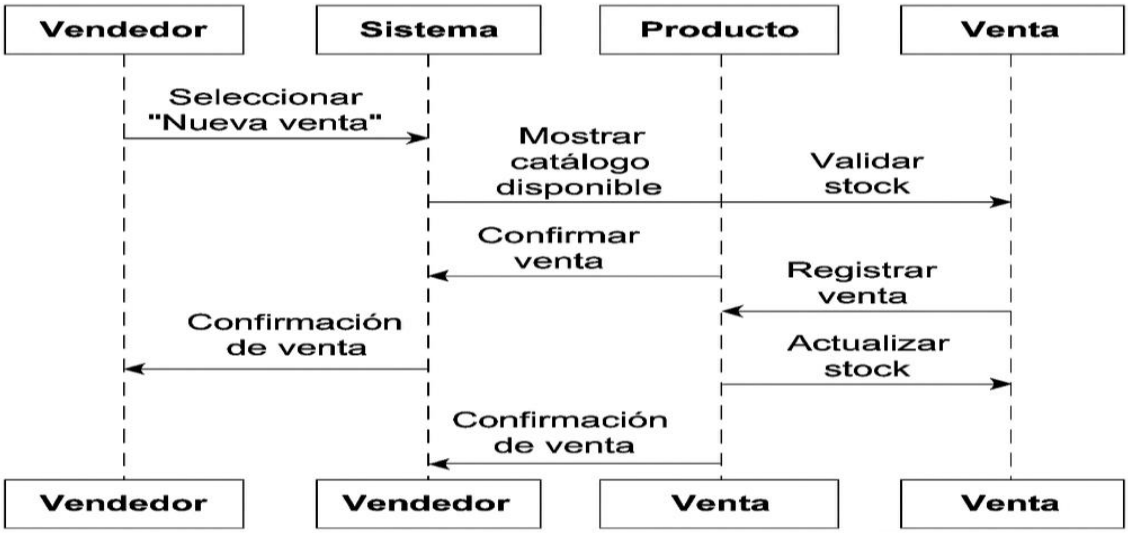
CU04- Generar Reportes

Diagrama de secuencia correspondiente al caso de uso: Generar Reportes.



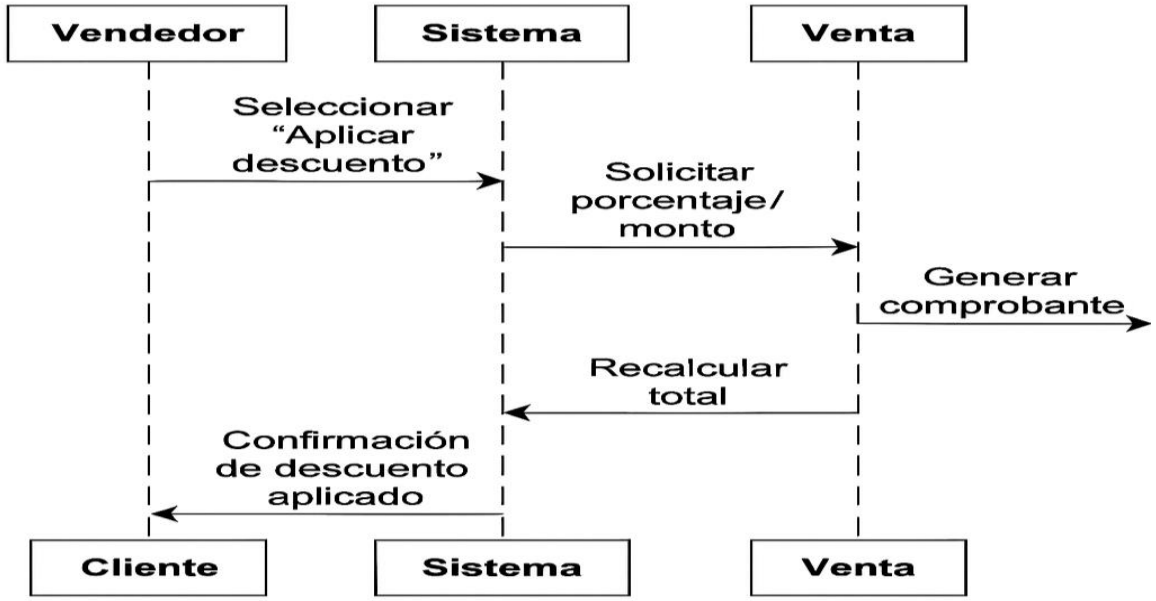
CU05- Realizar Venta

Diagrama de secuencia correspondiente al caso de uso: Realizar Venta.



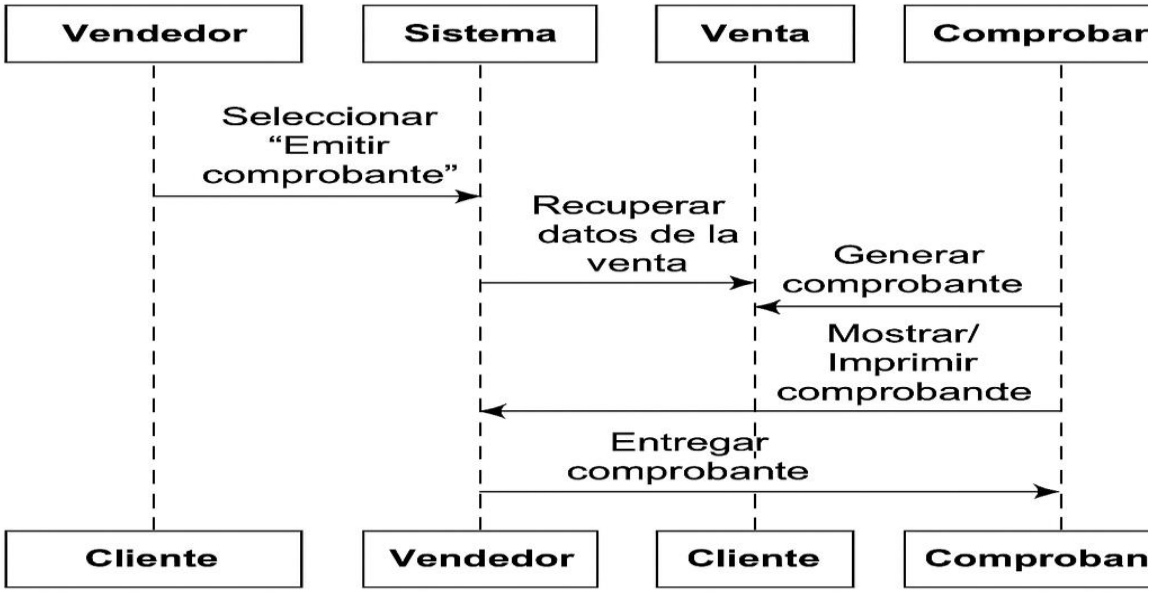
CU06- Aplicar Descuento

Diagrama de secuencia correspondiente al caso de uso: Aplicar Descuento.



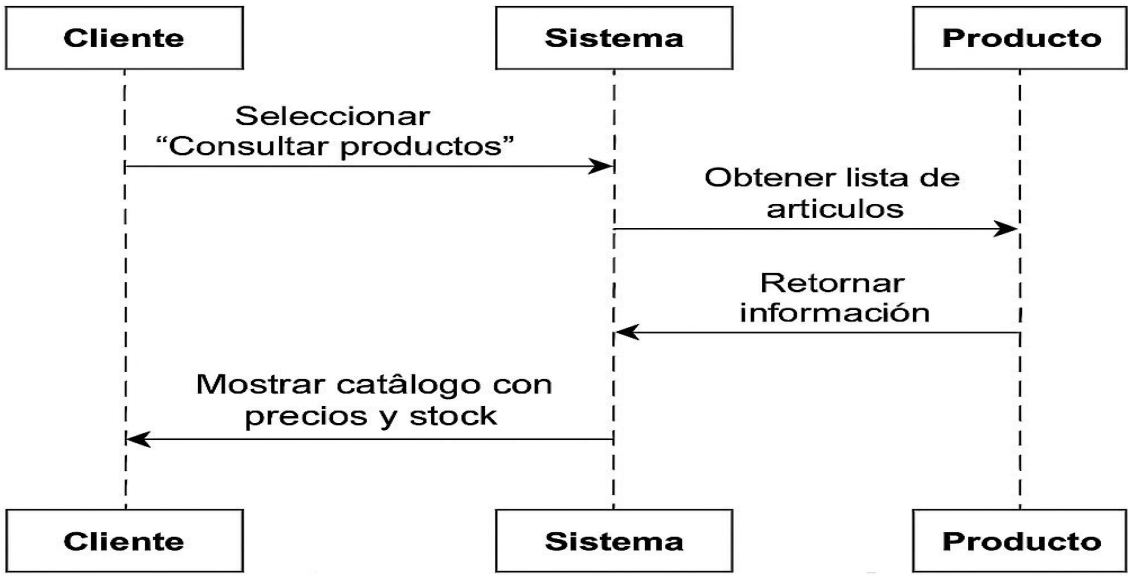
CU07- Emitir Comprobante

Diagrama de secuencia correspondiente al caso de uso: Emitir Comprobante.



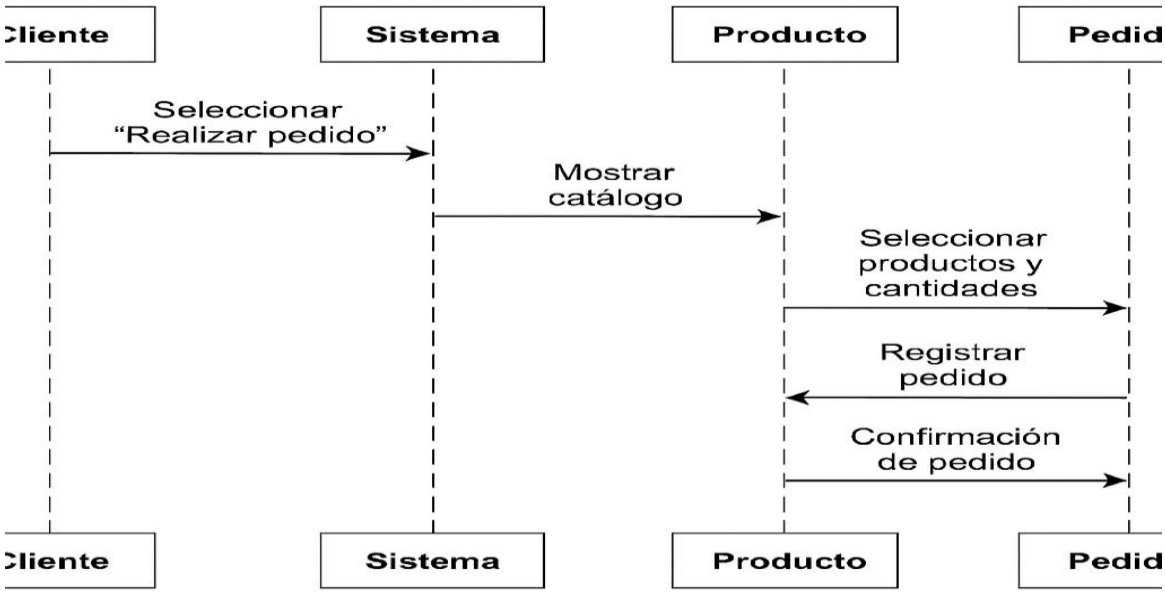
CU08- Consultar Productos

Diagrama de secuencia correspondiente al caso de uso: Consultar Productos.



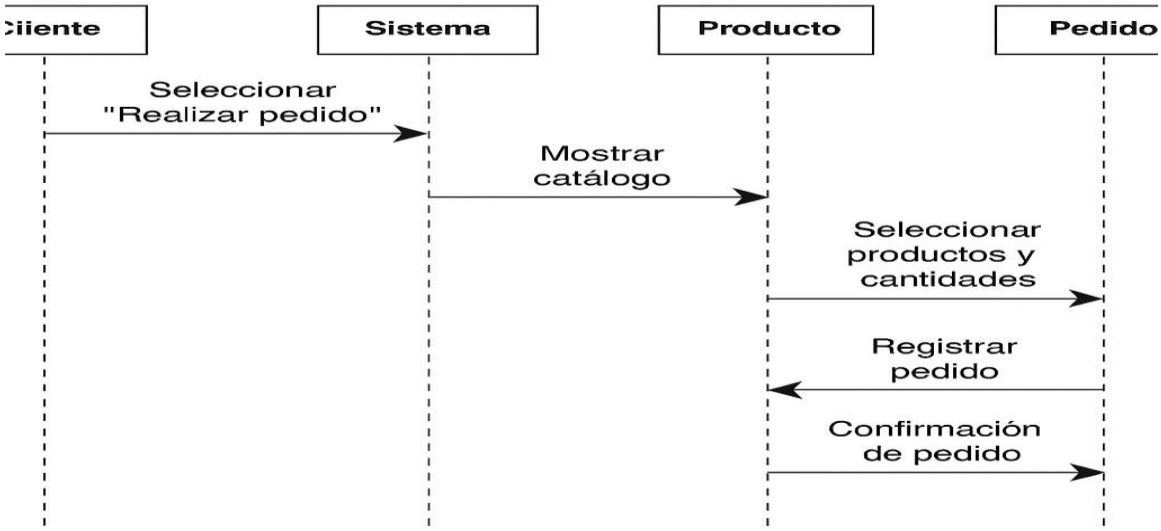
CU09- Realizar Pedido

Diagrama de secuencia correspondiente al caso de uso: Realizar Pedido.



CU10- Recibir Comprobante

Diagrama de secuencia correspondiente al caso de uso: Recibir Comprobante.



10. Etapa de Diseño

En la etapa de diseño se definió cómo será construido el sistema desde una perspectiva arquitectónica y de modelado.

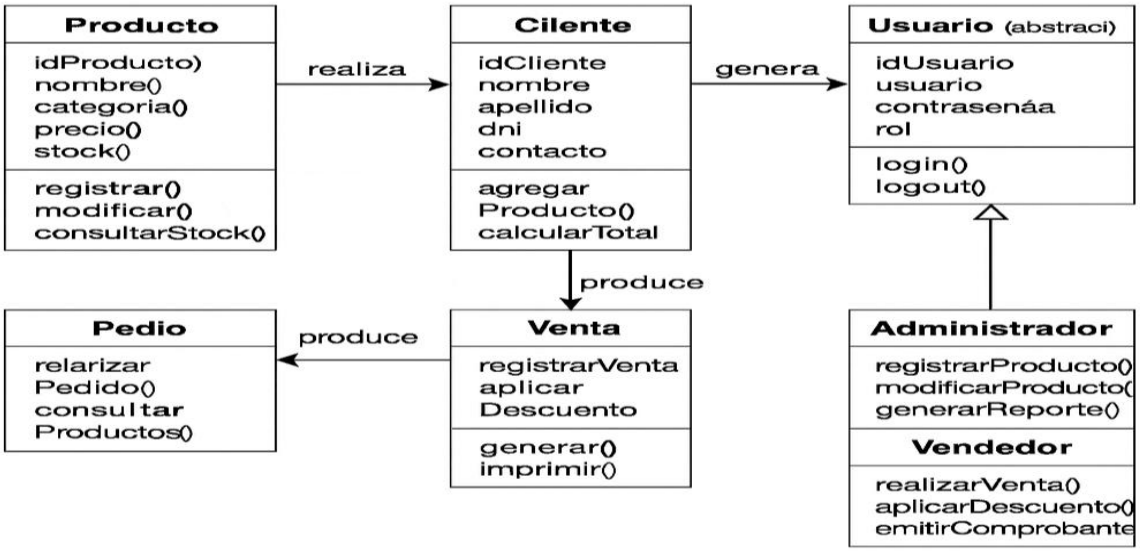
Se adoptó una arquitectura en **tres capas**:

- **Capa de presentación:** interfaz gráfica sencilla, orientada a la facilidad de uso por parte de administradores, vendedores y clientes.
- **Capa de negocio:** contiene las reglas principales del sistema, como la validación de stock, la aplicación de descuentos, la generación de reportes y el registro de ventas.
- **Capa de datos:** implementada sobre MySQL, garantiza la persistencia de la información de productos, clientes, pedidos, ventas y comprobantes.

Dentro del diseño también se elaboró el **diagrama de diseño** para la base de datos, que describe cómo se relacionan las entidades: un cliente realiza pedidos, los pedidos contienen productos, cada pedido genera una venta, y toda venta emite un comprobante.

El diseño buscó asegurar la modularidad, escalabilidad y mantenibilidad del sistema, facilitando futuras ampliaciones como el agregado de un módulo de compras a proveedores o la integración con plataformas de e-commerce.

Diagrama de diseño



Estructura destacada del diseño

Clases principales

Clase	Atributos clave	Métodos principales
Producto	idProducto, nombre, categoría, precio, stock	registrar(), modificar(), consultarStock()
Cliente	idCliente, nombre, apellido, dni, contacto	realizarPedido(), consultarProductos()
Pedido	idPedido, fecha, estado	agregarProducto(), calcularTotal()
Venta	idVenta, fecha, total, descuento	registrarVenta(), aplicarDescuento(), emitirComprobante()
Comprobante	idComprobante, fecha, tipo, detalle	generar(), imprimir()
Usuario *(abstracta)*	idUsuario, usuario, contraseña, rol	login(), logout()
Administrador	—	registrarProducto(), modificarProducto(), generarReporte()
Vendedor	—	realizarVenta(), aplicarDescuento(), emitirComprobante()

11. Etapa de Implementación

La implementación consiste en el desarrollo del sistema siguiendo el diseño planteado.

Para este proyecto se propone el uso de las siguientes tecnologías:

- **Lenguaje de programación:** Java o Python, debido a su robustez y disponibilidad de librerías para gestión de bases de datos.
- **Gestor de base de datos:** MySQL, por su estabilidad, escalabilidad y facilidad de uso.
- **Entorno de desarrollo:** NetBeans, Eclipse o VS Code.

Se implementarán los siguientes módulos:

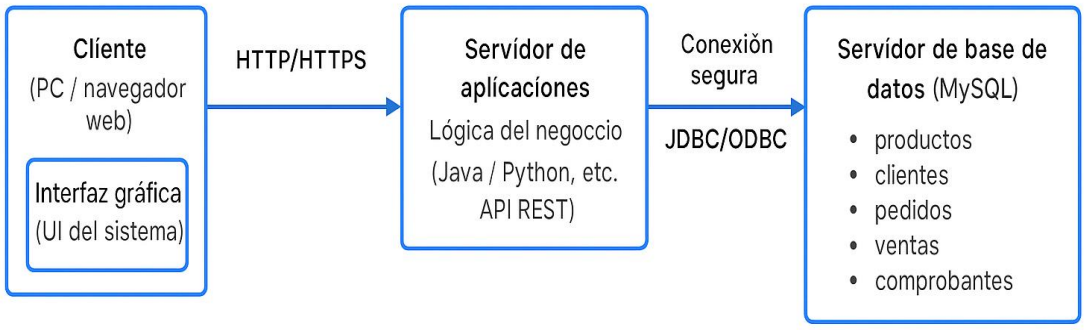
1. **Gestión de productos:** permite registrar, modificar, eliminar y consultar productos.
2. **Gestión de clientes:** permite registrar y mantener los datos de los clientes.
3. **Gestión de pedidos y ventas:** registra las operaciones comerciales, calcula totales y aplica descuentos.
4. **Emisión de comprobantes:** genera tickets o facturas asociadas a cada venta.
5. **Reportes:** consolida la información de ventas y stock para la toma de decisiones.
6. **Gestión de usuarios:** controla los accesos según el rol (administrador o vendedor).

Diagrama de Despliegue

Nodos principales:

- **Cliente (PC / Navegador Web)**
  - Interfaz gráfica (UI del sistema)
- **Servidor de Aplicaciones**
  - Lógica del negocio (Java/Python, API REST, etc.)
- **Servidor de Base de Datos (MySQL)**
  - Almacena información de productos, clientes, pedidos, ventas y comprobantes

Diagrama de Despliegue



12. Etapa de Pruebas

Una vez implementado, el sistema se someterá a un conjunto de pruebas que validen su correcto funcionamiento:

- **Pruebas unitarias:** cada módulo será verificado de manera aislada, por ejemplo, el registro de productos o la emisión de un comprobante.
- **Pruebas de integración:** se validará el flujo completo, como cuando un cliente realiza un pedido, este se convierte en una venta y se genera el comprobante.
- **Pruebas de aceptación:** se evaluará junto a los usuarios finales (administrador y vendedor) si el sistema satisface las necesidades y cumple con los requerimientos establecidos.

Estas pruebas garantizarán la confiabilidad, calidad y usabilidad de la solución antes de su puesta en producción.

Procedimiento general:

1. Preparación del entorno de pruebas (instalación del sistema, base de datos y usuarios de prueba).
2. Definición de los casos de prueba (funcionales y no funcionales).
3. Ejecución de los casos de prueba en cada módulo (productos, clientes, ventas, reportes).
4. Registro de resultados (éxito o error).
5. Tratamiento de defectos (corrección, re-prueba).
6. Informe final de validación.

Casos de Prueba Funcionales

Se enfocan en verificar que el sistema cumpla con lo que fue definido en los **requerimientos funcionales.**

ID	Funcionalidad	Caso de prueba	Resultado esperado
CF-01	Registrar producto	Registrar un producto con datos válidos	Producto guardado en la BD
CF-02	Registrar producto	Registrar producto con datos inválidos (ej. precio negativo)	Mensaje de error y no registra
CF-03	Modificar producto	Modificar datos de un producto existente	Datos actualizados correctamente
CF-04	Consultar stock	Consultar stock de productos	Muestra stock actual
CF-05	Generar reporte	Generar reporte de ventas en un rango de fechas válido	Reporte generado

ID	Funcionalidad	Caso de prueba	Resultado esperado
CF-06	Realizar venta	Registrar venta con productos disponibles	Venta guardada y stock actualizado
CF-07	Realizar venta	Intentar venta con stock insuficiente	Mensaje de error
CF-08	Aplicar descuento	Aplicar descuento válido sobre una venta	Total recalculado
CF-09	Emitir comprobante	Generar comprobante tras una venta	Comprobante emitido
CF-10	Consultar productos (Cliente)	Cliente consulta catálogo	Lista mostrada
CF-11	Realizar pedido (Cliente)	Cliente genera un pedido válido	Pedido guardado
CF-12	Recibir comprobante (Cliente)	Cliente recibe comprobante tras compra	Comprobante recibido

Casos de Prueba No Funcionales

Evalúan características de calidad: rendimiento, seguridad, usabilidad, etc.

ID	Categoría	Caso de prueba	Resultado esperado
CNF-01	Rendimiento	El sistema responde en menos de 3 segundos al consultar productos	Tiempo de respuesta < 3s
CNF-02	Rendimiento	El sistema soporta 50 usuarios concurrentes sin caídas	Flujo estable
CNF-03	Seguridad	Intentar acceder sin login	Acceso denegado
CNF-04	Seguridad	Validar encriptación de contraseñas en la BD	Contraseñas no legibles en BD
CNF-05	Usabilidad	Cliente navega por catálogo sin capacitación previa	Interfaz intuitiva
CNF-06	Compatibilidad	Emitir comprobante desde distintos navegadores (Chrome, Firefox, Edge)	Funciona en todos

ID	Categoría	Caso de prueba	Resultado esperado
CNF-07	Fiabilidad	Corte de red durante una venta	Venta queda en estado pendiente y no se pierde
CNF-08	Mantenibilidad	Modificar el precio de un producto desde el backend	Cambio reflejado sin afectar otros módulos

Tablero de Evaluación de Pruebas

Caso de prueba	Descripción	Resultado esperado	Resultado obtenido	Estado
CP-01 Registrar producto	El admin registra un producto con datos válidos	Producto registrado en BD	Correcto	Aprobado
CP-02 Registrar producto inval.	El admin ingresa precio negativo	Mensaje de error y no registra producto	Correcto	Aprobado
CP-03 Realizar venta	El vendedor selecciona productos con stock	Venta registrada y stock actualizado	Error en stock	Fallido
CP-04 Emitir comprobante	Se genera comprobante tras una venta	PDF impreso o descargable	Correcto	Aprobado
CP-05 Consultar productos	Cliente consulta catálogo	Lista de productos	Correcto	Aprobado

Tablero de Tratamiento de Defectos

ID Defecto	Descripción	Severidad	Estado	Responsable	Acción correctiva
D-01	Stock no se actualiza tras una venta	Alta	Pendiente	Dev Backend	Revisar trigger de actualización
D-02	No valida precio negativo en producto	Media	Resuelto	Dev Frontend	Agregar validación en formulario
D-03	El comprobante no se descarga en Safari	Baja	En progreso	Dev FullStack	Ajustar compatibilidad navegador

13. Definición de la Base de Datos para el sistema

El sistema requiere una **base de datos relacional** en **MySQL** que permita registrar y administrar la información de productos, clientes, usuarios, pedidos, ventas y comprobantes.

Entidades Principales

- **Producto**
  - idProducto (PK)
  - nombre
  - categoría
  - precio
  - stock
- **Cliente**
  - idCliente (PK)
  - nombre
  - apellido
  - dni
  - contacto
- **Usuario**
  - idUsuario (PK)
  - usuario
  - contraseña
  - rol (Administrador / Vendedor)
- **Pedido**
  - idPedido (PK)
  - fecha
  - estado
  - idCliente (FK)
- **DetallePedido** *(tabla intermedia entre Pedido y Producto)*
  - idDetalle (PK)
  - idPedido (FK)
  - idProducto (FK)
  - cantidad
  - subtotal
- **Venta**
  - idVenta (PK)
  - fecha
  - total
  - descuento
  - idPedido (FK)
  - idUsuario (FK)
- **Comprobante**
  - idComprobante (PK)
  - fecha
  - tipo (Factura, Ticket)
  - idVenta (FK)

Relaciones Principales

- **Un Cliente** puede realizar **muchos Pedidos.**
- **Un Pedido** contiene **muchos Productos** (a través de DetallePedido).
- **Un Pedido** puede generar **una Venta.**
- **Una Venta** produce **un Comprobante.**
- **Un Usuario (Administrador/Vendedor)** gestiona ventas y productos.

Justificación del diseño

- Se usa **DetallePedido** para resolver la relación *muchos a muchos* entre Pedido y Producto.
- Se separan **Usuario y Cliente**, ya que cumplen roles distintos en el sistema.
- La entidad **Comprobante** se asocia a **Venta**, lo que garantiza trazabilidad y soporte para facturación.
- La normalización se aplica hasta la **3FN (Tercera Forma Normal),** evitando redundancia de datos.

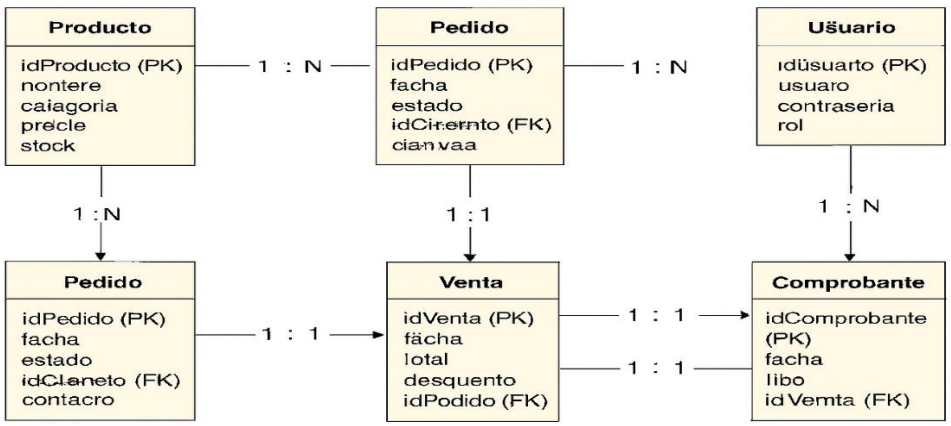
14. Diagrama Entidad-Relación (DER)

El siguiente diagrama Entidad-Relación (DER) representa las entidades principales del **Sistema de Gestión de Ventas de Artículos Escolares**, sus atributos y las relaciones entre ellas. Este diseño garantiza la consistencia e integridad de los datos, permitiendo un modelo normalizado hasta la tercera forma normal (3FN).

Explicación del diagrama:

- Un **Cliente** puede realizar muchos Pedidos.
- Un **Pedido** puede contener muchos Productos (a través de DetallePedido).
- Un **Pedido** genera una Venta, y cada Venta produce un Comprobante.
- Los **Usuarios** (Administradores y Vendedores) están vinculados a las Ventas.

Este modelo refleja fielmente el flujo del negocio desde el registro de pedidos hasta la emisión de comprobantes, asegurando trazabilidad completa.



Este diagrama es la base para la creación de las tablas en MySQL.

Entidades y atributos clave

1. **Producto**
   - idProducto (PK)
   - nombre
   - categoria
   - precio
   - stock
     - 🔻 Un producto puede estar en muchos pedidos.
2. **Cliente**
   - idCliente (PK)
   - nombre
   - apellido
   - dni
   - contacto
     - 🔻 Un cliente puede realizar muchos pedidos.
3. **Usuario** *(Administrador / Vendedor)*
   - idUsuario (PK)
   - usuario
   - contraseña
   - rol
     - 🔻 Un usuario puede registrar ventas y gestionar productos.
4. **Pedido**
   - idPedido (PK)
   - fecha
   - estado
   - idCliente (FK) → Cliente.idCliente
     - 🔻 Cada pedido pertenece a un cliente.
5. **DetallePedido** *(resuelve muchos a muchos entre Pedido y Producto)*
   - idDetalle (PK)
   - idPedido (FK) → Pedido.idPedido
   - idProducto (FK) → Producto.idProducto
   - cantidad
   - subtotal
6. **Venta**
   - idVenta (PK)
   - fecha
   - total
   - descuento
   - idPedido (FK) → Pedido.idPedido
   - idUsuario (FK) → Usuario.idUsuario
     - 🔻 Cada venta está vinculada a un pedido y realizada por un usuario (vendedor).
7. **Comprobante**
   - idComprobante (PK)
   - fecha

- tipo (Factura, Ticket)
- idVenta (FK) → Venta.idVenta
  ⬘ Cada venta produce un comprobante.

Relaciones y cardinalidades

- **Cliente – Pedido**
  - 1 Cliente puede realizar *muchos* pedidos.
  - 1 Pedido pertenece a *un solo* cliente.
    → (1:N)
- **Pedido – DetallePedido – Producto**
  - 1 Pedido puede contener *muchos* productos.
  - 1 Producto puede estar en *muchos* pedidos.
    → Relación (N:M) resuelta con **DetallePedido**.
- **Pedido – Venta**
  - 1 Pedido puede generar *una sola* venta.
  - 1 Venta está vinculada a *un solo* pedido.
    → (1:1)
- **Venta – Comprobante**
  - 1 Venta produce *un comprobante*.
  - 1 Comprobante pertenece a *una sola* venta.
    → (1:1)
- **Usuario – Venta**
  - 1 Usuario puede registrar *muchas* ventas.
  - 1 Venta siempre es registrada por *un usuario* (vendedor).
    → (1:N)

Ventajas del DER propuesto

- El uso de **DetallePedido** asegura integridad y flexibilidad en pedidos con múltiples productos.
- La relación **Pedido–Venta–Comprobante** refleja el flujo real del negocio.
- Separar **Cliente** de **Usuario** permite diferenciar claramente los roles de consumo y gestión.
- La base está normalizada hasta la **Tercera Forma Normal (3FN)**, evitando redundancia de datos y mejorando consistencia.

15. Creación de las Tablas en MySQL

```sql
CREATE TABLE Producto (
  id_producto INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  categoria VARCHAR(50),
  precio DECIMAL(10,2) NOT NULL,
  stock INT NOT NULL
);

CREATE TABLE Cliente (
  id_cliente INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100),
  apellido VARCHAR(100),
  dni VARCHAR(20),
  contacto VARCHAR(50)
);
```

```sql
CREATE TABLE Pedido (
  id_pedido INT AUTO_INCREMENT PRIMARY KEY,
  id_cliente INT,
  fecha DATE,
  estado VARCHAR(20),
  FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente)
);

CREATE TABLE Venta (
  id_venta INT AUTO_INCREMENT PRIMARY KEY,
  id_pedido INT,
  fecha DATE,
  total DECIMAL(10,2),
  FOREIGN KEY (id_pedido) REFERENCES Pedido(id_pedido)
);
```

```sql
CREATE TABLE Comprobante (
  id_comprobante INT AUTO_INCREMENT PRIMARY KEY,
  id_venta INT,
  tipo VARCHAR(20),
  fecha DATE,
  monto DECIMAL(10,2),
  FOREIGN KEY (id_venta) REFERENCES Venta(id_venta)
);

CREATE TABLE Usuario (
  id_usuario INT AUTO_INCREMENT PRIMARY KEY,
  nombre_usuario VARCHAR(50) UNIQUE,
  contraseña VARCHAR(255) NOT NULL,
  rol ENUM('Administrador','Vendedor') NOT NULL
);
```

Resultados en tablas:

Tabla: Producto

id producto	nombre	categoría	precio	stock
1	Lápiz HB	Escritura	15.00	120
2	Cuaderno A4	Papelería	120.00	60
3	Mochila escolar	Mochilas	3500.00	25

Tabla: Cliente

id cliente	nombre	apellido	dni	contacto
1	Laura	Gómez	30123456	laura@gmail.com
2	Martín	Pérez	28987654	1122334455

Tabla: Usuario

id usuario	nombre usuario	contraseña (hash)	rol
1	admin01	2b$10...	Administrador
2	vendedor02	2b$10...	Vendedor

Tabla: Pedido

id pedido	id cliente	fecha	estado
1	1	2025-10-01	Pendiente
2	2	2025-10-02	Confirmado

Tabla: Venta

id venta	id pedido	fecha	total
1	2	2025-10-02	3620.00

Tabla: Comprobante

id comprobante	id venta	tipo	fecha	monto
1	1	Factura	2025-10-02	3620.00

16. Inserción, Consulta y Borrado de Registros

```
-- Inserción de productos
INSERT INTO Producto (nombre, categoria, precio, stock)
VALUES ('Cuaderno', 'Papelería', 250.00, 50);
```

Resultado esperado en la tabla Producto:

id producto	nombre	categoría	precio	stock
1	Cuaderno	Papelería	250.00	50

(Suponiendo que es el primer registro y el id producto *se autogenera como 1.)*

```
-- Consulta de stock
SELECT nombre, stock FROM Producto;
```

Resultado de la consulta:

nombre	stock
Cuaderno	50

```
-- Borrado de un cliente
DELETE FROM Cliente WHERE id_cliente = 3;
```

Resultado esperado:

- Si el cliente con id cliente = 3 existe, se elimina de la tabla.
- Si no existe, no se realiza ningún cambio.

17. Presentación de Consultas SQL

https://github.com/cristianarandaeuge7-boop/consulta-SQL/blob/5f5f4cfd4229023f5d0915ae908d73c0cbb30bb5/.gitattributes

Requerimientos de Comunicación del Sistema

Entorno de red

- Topología Cliente-Servidor.
- Los usuarios acceden al Servidor de Aplicaciones.
- El Servidor de Aplicaciones interactúa con la Base de Datos MySQL en red local segura.

Infraestructura física

- Servidor de Aplicaciones: alojado en Linux/Windows Server.
- Servidor de Base de Datos: conectado en la misma LAN.
- Estaciones de trabajo: PCs o notebooks conectadas vía Ethernet o Wi-Fi.

Protocolos de comunicación

- **Cliente ↔ Servidor de Aplicaciones:** HTTP/HTTPS (se recomienda SSL/TLS).
- **Servidor de Aplicaciones ↔ Base de Datos:** JDBC/ODBC con protocolo MySQL cifrado.
- **Sistemas externos (si aplica):** REST API con JSON/XML y autenticación OAuth2 o JWT.

Control de enlace de datos

- Comunicación LAN mediante Ethernet (IEEE 802.3).
- Conexiones Wi-Fi mediante IEEE 802.11ac con WPA2/WPA3.
- Para acceso remoto, se recomienda VPN.

Estándares de seguridad aplicados

- Autenticación de usuarios con contraseñas cifradas (bcrypt o SHA-256).
- Cifrado de datos en tránsito mediante TLS 1.2/1.3.
- Control de accesos mediante roles diferenciados (Administrador, Vendedor, Cliente).

18. Definiciones de Comunicación

El sistema se comunica de la siguiente forma:

- Los **usuarios internos (administrador y vendedor)** interactúan mediante una interfaz de usuario.
- La **lógica del negocio** procesa las operaciones (validación de stock, cálculo de descuentos, registro de ventas).
- La **base de datos MySQL** almacena y gestiona la información.

La comunicación entre la aplicación y la base de datos se realiza a través de **conectores seguros**. Si existiera acceso remoto, se recomienda el uso de **protocolos cifrados (SSL/TLS)** para garantizar la seguridad de los datos.

Entorno de red y arquitectura

- **Topología Cliente–Servidor**: todos los usuarios acceden al sistema desde clientes ligeros (PC, notebook, móvil) que se conectan al Servidor de Aplicaciones.
- **Servidor de Aplicaciones** gestiona la lógica del negocio y sirve de intermediario entre los clientes y la base de datos.
- **Servidor de Base de Datos** (MySQL) dedicado, con acceso restringido solo desde el servidor de aplicaciones.
- **Posibilidad de nube**: el sistema podría desplegarse en un servicio cloud (AWS, Azure, GCP) para escalabilidad.

Infraestructura física y lógica

- **Servidor de Aplicaciones**:
  - CPU de 4 núcleos, 16 GB RAM, 200 GB SSD.
  - Sistema operativo Linux (Ubuntu Server) o Windows Server.
  - Contenedor de aplicaciones (Tomcat, Flask/Django o Node.js).
- **Servidor de Base de Datos**:
  - 8 núcleos, 32 GB RAM, 500 GB SSD (RAID 10).
  - Motor MySQL 8.0.
  - Replicación maestra–esclavo para alta disponibilidad.
- **Clientes**: navegadores web modernos (Chrome, Firefox, Edge) o aplicación de escritorio ligera.

Protocolos de comunicación y estándares

- **Cliente ↔ Servidor de Aplicaciones:**
  - HTTP/2 o HTTP/3 sobre **HTTPS** con certificados SSL/TLS.
  - Compresión de datos GZIP para optimizar transferencia.
- **Servidor de Aplicaciones ↔ Base de Datos:**
  - JDBC/ODBC.
  - Conexiones seguras con **SSL cifrado**.
  - Mecanismos de *connection pooling* para eficiencia.

- **Comunicación con sistemas externos (integraciones futuras):**
  - REST API con JSON o XML.
  - Estándares de autenticación: OAuth 2.0 o JWT.
  - Posible integración con servicios de facturación electrónica.

Seguridad en la comunicación

- Autenticación de usuarios con contraseñas cifradas (bcrypt o SHA-256).
- Roles y permisos diferenciados (Administrador, Vendedor, Cliente).
- Monitoreo de logs de conexión y auditoría de transacciones.
- Firewalls perimetrales e internos para segmentar la red.
- Acceso remoto solo vía **VPN cifrada**.

Control de enlace y confiabilidad

- **LAN:** Ethernet (IEEE 802.3).
- **WLAN:** Wi-Fi (IEEE 802.11ac/ax) con WPA3.
- **QoS (Quality of Service):** prioridad para tráfico de base de datos y servicios críticos.
- **Alta disponibilidad:** balanceadores de carga (HAProxy, Nginx) para múltiples servidores de aplicación.
- **Backup y recuperación:** copias de seguridad automáticas en servidor secundario y en la nube.

Escalabilidad y futuro

- Arquitectura diseñada para soportar **escalamiento horizontal** (añadiendo más servidores de aplicación).
- Separación de micro servicios en caso de crecimiento (ejemplo: módulo de facturación, módulo de reportes).
- Preparado para migración a contenedores con Docker/Kubernetes.

19. Explicación del desarrollo en Java

1. Introducción general del desarrollo

El sistema fue implementado utilizando el lenguaje **Java**, elegido por su robustez, portabilidad y amplio soporte para el desarrollo de aplicaciones orientadas a objetos. El objetivo principal del desarrollo fue construir una aplicación modular, fácil de mantener y extensible, capaz de gestionar las operaciones de ventas, stock y reportes de una librería de artículos escolares.

El enfoque se basó en una arquitectura en tres capas:

- **Capa de presentación:** interfaz de usuario y menú de opciones.
- **Capa de negocio:** lógica de la aplicación (gestión de productos, ventas, usuarios, reportes).
- **Capa de datos:** conexión con la base de datos MySQL.

2. Correcta utilización de sintaxis, tipos de datos y estructuras de control

Durante el desarrollo se aplicaron:

- **Tipos de datos primitivos y objetos**, como *int, double, String, ArrayList<>,* etc.
- **Estructuras de control** (*if, switch, for, while, do-while*) para la navegación de menús, validaciones de entrada y recorridos de colecciones.
- **Colecciones genéricas (***ArrayList***,** *HashMap***)** para almacenar y manipular listas de productos, clientes o ventas.

Ejemplo simplificado:

```
for (Producto p : listaProductos) {
    if (p.getStock() < 10) {
        System.out.println("Stock bajo: " + p.getNombre());
    }
}
```

3. Manejo de excepciones

El sistema implementa bloques ***try-catch*** para controlar posibles errores durante la ejecución, como entradas inválidas del usuario o fallas de conexión a la base de datos.
Esto asegura que el programa no se detenga abruptamente ante un error.

Ejemplo:

```
try {
    conexion = DriverManager.getConnection(url, usuario, contraseña);
} catch (SQLException e) {
    System.out.println("Error al conectar con la base de datos: " + e.getMessage());
}
```

4. Aplicación de principios de Programación Orientada a Objetos

El sistema aplica los cuatro pilares de la POO:

a) Encapsulamiento

Todos los atributos de las clases son **privados**, con acceso mediante **métodos getters y setters**, lo que protege los datos de modificaciones externas.

```
public class Producto {
    private String nombre;
    private double precio;
    private int stock;

    public double getPrecio() { return precio; }
    public void setPrecio(double precio) { this.precio = precio; }
}
```

b) Herencia

Se implementa una jerarquía de clases para los usuarios del sistema:

```
public abstract class Usuario {
    protected String usuario;
    protected String contraseña;
    protected String rol;
}

public class Administrador extends Usuario {
    public void generarReporte() { ... }
}

public class Vendedor extends Usuario {
    public void realizarVenta() { ... }
}
```

c) Polimorfismo

Las subclases (*Administrador, Vendedor*) pueden sobrescribir métodos definidos en la clase base Usuario, adaptando su comportamiento según el rol.

d) Abstracción

Se definen clases abstractas y métodos que representan comportamientos generales (por ejemplo, *Usuario*), ocultando detalles de implementación y mostrando solo lo esencial para su uso.

5. Menú de selección

El sistema presenta un menú interactivo en consola (o interfaz gráfica, según versión), mediante estructuras repetitivas y condicionales, permitiendo al usuario elegir entre distintas operaciones.

Ejemplo:

```
int opcion;
do {
    System.out.println("1. Registrar producto");
    System.out.println("2. Realizar venta");
    System.out.println("3. Consultar stock");
    System.out.println("0. Salir");
    opcion = scanner.nextInt();

    switch(opcion) {
        case 1: registrarProducto(); break;
        case 2: realizarVenta(); break;
        case 3: consultarStock(); break;
    }
} while(opcion != 0);
```

6. Creación de objetos y constructores

Cada entidad del sistema (Producto, Cliente, Venta, Comprobante, etc.) se instancia utilizando **constructores** que inicializan los atributos principales.

```
Producto p1 = new Producto("Cuaderno A4", "Papelería", 250.00, 50);
Venta v1 = new Venta(p1, 3, new Date());
```

Desarrollo en Java – Clases y Métodos Implementados

Clase Producto

Encapsula los datos de cada artículo escolar y gestiona su stock.

```java
public class Producto {
    private int idProducto;
    private String nombre;
    private String categoria;
    private double precio;
    private int stock;

    // Constructor
    public Producto(int idProducto, String nombre, String categoria, double precio, int stock) {
        this.idProducto = idProducto;
        this.nombre = nombre;
        this.categoria = categoria;
        this.precio = precio;
        this.stock = stock;
    }

    // Métodos getters y setters (Encapsulamiento)
    public String getNombre() { return nombre; }
    public void setNombre(String nombre) { this.nombre = nombre; }

    public double getPrecio() { return precio; }
    public void setPrecio(double precio) { this.precio = precio; }

    public int getStock() { return stock; }
    public void setStock(int stock) { this.stock = stock; }

    // Métodos de lógica
    public void actualizarStock(int cantidadVendida) {
```

```java
    public int getStock() { return stock; }
    public void setStock(int stock) { this.stock = stock; }

    // Métodos de lógica
    public void actualizarStock(int cantidadVendida) {
        if (cantidadVendida > stock) {
            System.out.println("Stock insuficiente");
        } else {
            stock -= cantidadVendida;
        }
    }

    public void mostrarInfo() {
        System.out.println(nombre + " - $" + precio + " (Stock: " + stock + ")");
    }
}
```

Clase Cliente

Representa a las personas que compran en la librería.

```
public class Cliente {
    private int idCliente;
    private String nombre;
    private String apellido;
    private String dni;
    private String contacto;

    public Cliente(int idCliente, String nombre, String apellido, String dni, String contacto) {
        this.idCliente = idCliente;
        this.nombre = nombre;
        this.apellido = apellido;
        this.dni = dni;
        this.contacto = contacto;
    }

    public String getNombreCompleto() {
        return nombre + " " + apellido;
    }
}
```

Clase abstracta **Usuario**

Base para los roles de Administrador **y** Vendedor**.**
Aplica abstracción **y** herencia**.**

```
public abstract class Usuario {
    protected String usuario;
    protected String contraseña;
    protected String rol;

    public Usuario(String usuario, String contraseña, String rol) {
        this.usuario = usuario;
        this.contraseña = contraseña;
        this.rol = rol;
    }

    public abstract void mostrarMenu(); // Método abstracto
}
```

Subclase Administrador

Hereda de Usuario y aplica polimorfismo**.**

```java
public class Administrador extends Usuario {

    public Administrador(String usuario, String contraseña) {
        super(usuario, contraseña, "Administrador");
    }

    @Override
    public void mostrarMenu() {
        System.out.println("1. Registrar producto");
        System.out.println("2. Modificar producto");
        System.out.println("3. Generar reportes");
    }

    public void generarReporte() {
        System.out.println("Generando reporte de ventas...");
    }
}
```

Subclase Vendedor

Implementa funciones específicas del punto de venta.

```java
public class Vendedor extends Usuario {

    public Vendedor(String usuario, String contraseña) {
        super(usuario, contraseña, "Vendedor");
    }

    @Override
    public void mostrarMenu() {
        System.out.println("1. Realizar venta");
        System.out.println("2. Aplicar descuento");
        System.out.println("3. Emitir comprobante");
    }

    public void realizarVenta(Producto p, int cantidad) {
        try {
            p.actualizarStock(cantidad);
            System.out.println("Venta realizada correctamente");
        } catch (Exception e) {
            System.out.println("Error al procesar la venta: " + e.getMessage());
        }
    }
}
```

Clase Venta

Maneja la operación de compra y sus totales.

```java
public class Venta {
    private int idVenta;
    private Date fecha;
    private double total;

    public Venta(int idVenta, Date fecha, double total) {
        this.idVenta = idVenta;
        this.fecha = fecha;
        this.total = total;
    }

    public void aplicarDescuento(double porcentaje) {
        total = total - (total * porcentaje / 100);
    }

    public void emitirComprobante() {
        System.out.println("Comprobante emitido por $" + total);
    }
}
```

20. Presentación del desarrollo en Java.

Introducción del sistema

- **Nombre del proyecto**: Sistema de Gestión de Ventas de Artículos Escolares
- **Objetivo**: Automatizar la gestión de productos, usuarios y ventas en una librería escolar.
- **Tecnologías utilizadas**:

- Java (VS Code)
- MySQL
- JDBC
- Git y GitHub

Estructura del sistema

Clases principales (8 en total):

Main	
SistemaGestion	
Usuario	
Administrador	Usuario
Vendedor	Usuario
Producto	
Venta	
ConexionBD	

Conexión con la base de datos

- Clase Conexión BD encapsula la conexión con MySQL.
- Base de datos: librería
- Tablas creadas: productos, ventas, usuarios (simulada)
- Probado con clase TestConexion.java

Organización del proyecto

- Subido a GitHub: https://github.com/cristianarandaeuge7-boop/GestionDeVentasArticulosEscolares.git
- Incluye: gitignore y README.md
- Código limpio, modular y orientado a objetos


Administrador.java


Cliente.java


Main.java


Producto.java


SistemaGestion.java


Vendedor.java


Venta.java


Usuario.java

Uso complementario de arreglos y ArrayList

En el proyecto se implementa un `ArrayList<Producto>` para gestionar la colección dinámica de productos dentro del sistema. Esta estructura permite agregar y manipular objetos de forma flexible, lo que resulta adecuado para modelos de datos que pueden crecer durante la ejecución.

Complementariamente, se incorpora un arreglo (array) de tipo String para almacenar categorías predefinidas de productos. Este arreglo representa una estructura de datos de tamaño fijo y permite ejemplificar el uso de colecciones tradicionales en Java.

```
private String[] categoriasPredefinidas = {
    "Útiles",
    "Mochilas",
    "Arte",
    "Tecnología",
    "Papelería"
};
```

Video explicativo

https://youtube.com/shorts/QloYVaYVs2M?si=RPLoDkxBpqFt0iC9